The Graduate Journey: Education and Labour Market Realities in Canada

Fuxi Ma fuxi.ma@outlook.com

Table of contents

1 National Graduates Survey- class of 2020 (Data collected in 2023)

```
import pandas as pd
import seaborn as sns
import matplotlib as plt
from IPython.display import display, Markdown

# Read the CSV file
try:
    # Read the CSV file into a pandas DataFrame
    df = pd.read_csv('ngs2020.csv')

    # Display basic information about the dataset
    display(Markdown("<span style='color: green'>Dataset information:</span>"))
    print(f"Number of rows: {df.shape[0]}")
    print(f"Number of columns: {df.shape[1]}\n")
    df.info()
    print("\n")
    display(Markdown("<span style='color: green'>Column names:</span>"))
```

```
    print(" ".join(list(df.columns)),"\n")

    # Number of missing data
    missing_data = df.isnull().sum().sum()
    if missing_data == 0:
        print(f"\033[30;43mThere are no missing data.\033[0m")
    else:
        print(f"\033[30;43mThere are {missing_data} missing data.\033[0m")

except FileNotFoundError:
    print("Error: The file 'ngs2020.csv' was not found in the current directory.")
except pd.errors.EmptyDataError:
    print("Error: The file 'ngs2020.csv' is empty.")
except pd.errors.ParserError:
    print("Error: There was an issue parsing the CSV file. Check if it's properly formatted.
```

Dataset information:

```
Number of rows: 16138
Number of columns: 114

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16138 entries, 0 to 16137
Columns: 114 entries, PUMFID to DDIS_FL
dtypes: int64(114)
memory usage: 14.0 MB
```

Column names:

```
PUMFID CERTLEVP REG_INST REG_RESP PGMCIPAP PGM_P034 PGM_P036 PGM_P100 PGM_P111 PGM_280A PGM_

There are no missing data.
```

2 NGS Questions

```
import yaml
import os

# Path to the YAML file
file_path = 'ngs2020_questions.yaml'

try:
    # Open and load the YAML file
```

```
    with open(file_path, 'r') as file:
        questions = yaml.safe_load(file)

    # Print the loaded question structure

    print(f'\033[32m\nPUMFID: \033[0m Public Use Microdata File ID - {questions["PUMFID"]}\n
    print(f"Questions ({len(questions)-1}):\n")
    for question in questions:
        if question != 'PUMFID':
            print(f'\033[32m{question}: \033[0m {questions[question]}')

except FileNotFoundError:
    print(f"Error: File '{file_path}' not found.")
except yaml.YAMLError as e:
    print(f"Error parsing YAML file: {e}")
```

```
PUMFID:  Public Use Microdata File ID - Randomly generated record identifier for the PUMF fi

Questions (113):

CERTLEVP:  2020 Program - Level of study - Grouping
REG_INST:  2020 Program - Region of postsecondary educational institution
REG_RESP:  Time of interview 2023 - Region of primary residence
PGMCIPAP:  2020 Program - Aggregated CIP 2021
PGM_P034:  2020 Program - Full-time or part-time student
PGM_P036:  2020 Program - Reason did not take program full-time
PGM_P100:  Work placement during program
PGM_P111:  Work placement during prog - Description
PGM_280A:  Entrepreneurial skills - Started a business
PGM_280B:  Entrepreneurial skills - Completed courses
PGM_280C:  Entrepreneurial skills - Business plan or pitch competition
PGM_280D:  Entrepreneurial skills - Visited an entrepreneurship centre
PGM_280E:  Entrepreneurial skills - Worked on an entrepreneurship project
PGM_280F:  Entrepreneurial skills - None of the above
PGM_290:  2020 Program - Worked during program
PGM_350:  2020 Program - Volunteer activities during program
PGM_380:  2020 Program - Components taken outside of Canada
PGM_P401:  2020 Program - Online or distance education
PGM_410:  2020 Program - Main factor in choice of postsecondary institution
PGM_415:  2020 Program - Main factor in choice of program
PGM_430:  2020 Program - Choose the same field of study or specialization again
COV_010:  COVID-19 - Completion of program delayed
```

COV_070: COVID-19 - Plans for further postsecondary education changed
COV_080: COVID-19 - Employment status/plans affected
EDU_010: After 2020 program - Other postsecondary programs taken
EDU_P020: After 2020 program - Number of other programs taken
HLOSINTP: Time of interview 2023 - Aggregated highest level of ed. completed
STL_010: Government-student loan program - Applied
STL_020: Government-student loan program - Applications approved
STULOANS: Government-student loan program - Received
STL_030: Government-student loan program - Main reason did not apply
OWESLGD: Government-student loan program - Debt size - Graduation 2020
OWEGVIN: Government-student loan program - Debt size - Interview 2023
STL_080: Government-student loan program - Remission/debt reduction/loan forg.
STL_100A: Received government assistance: Repayment assistance plan
STL_100B: Received government assistance: Revision of terms
STL_100C: Received government assistance: Interest only payments
STL_100D: Received government assistance: None of the above
STL_130: Government-student loan program - Total repayment term
STL_150: Government-student loan program - Repaymt of loan from financial inst.
STL_160B: Sources of funding - RESP
STL_160C: Sources of funding - Government grants or bursaries
STL_160D: Sources of funding - Non-government grants or bursaries
STL_160E: Sources of funding - Scholarships or awards
STL_160F: Sources of funding - Employment earnings or savings
STL_160G: Sources of funding - Research or teaching assistantship
STL_160H: Sources of funding - Parents, family, friends
STL_160I: Sources of funding - Bank or institution loans
STL_160J: Sources of funding - Credit cards
STL_160L: Sources of funding - Employer
STLP160N: Sources of funding - Other
SRCFUND: Sources of funding - Number of sources - All postsecondary edu
STL_170A: Main source of funding - Government student loans
STL_170B: Main source of funding - RESP
STL_170C: Main source of funding - Government grants or bursaries
STL_170D: Main source of funding - Non-government grants or bursaries
STL_170E: Main source of funding - Scholarships or awards
STL_170F: Main source of funding - Employment earnings or savings
STL_170G: Main source of funding - Research or teaching assistantship
STL_170H: Main source of funding - Parents, family, friends
STL_170I: Main source of funding - Bank or institution loans
STL_170J: Main source of funding - Credit cards
STL_170L: Main source of funding - Employer
STLP170N: Main source of funding - Other
RESPP: RESP - Total amount received for postsecondary education

STL_190: Repay loans from family or friends for education
DBTOTGRD: Loans at graduation 2020 - Debt size of non-government loans (range)
DBTALGRD: Loans at graduation 2020 - Debt size of all loans
DBTOTINT: Time of interview 2023 - Debt size of non-government loans (range)
DBTALINT: Time of interview 2023 - Debt size of all loan
SCHOLARP: Total amount received from scholarships/awards/fellowships and prizes
LMA_010: Reference week - Attended school, college, CEGEP or university
LFSTATP: Reference week - Labour force status
LMA2_07: Reference week - More than one job or business
LMA3_P01: Reference week - Employee or self-employed
LFCINDP: Reference week - Sector for job
LFCOCCP: Reference week - Broad occupational category for job
LFWFTPTP: Reference week - Full-time or part-time status of job or business
LMA6_05: Reference week - Job permanent or not permanent
LMA6_08: Reference week - Main method used to find job
JOBQLEVP: Reference week - Aggregated level of studies required to get job
JOBQLGRD: Reference week - Qualification for job compared to 2020 program
JOBQLINT: Reference week - Qualification job vs level of education
LMA6_11: Reference week - Relatedness of job or business to 2020 program
LMA6_12: Reference week - Qualification level for job
LMA6_13A: Reference week - Satisfied with overall job
LMA6_13B: Reference week - Satisfied with wage or salary of job
LMA6_13C: Reference week - Satisfied with job security
JOBINCP: Reference week - Annual wage or salary for job
LMA6_15: After program 2020 - First job
AFT_P010: After 2020 program - Number of jobs or businesses
AFT_P020: After 2020 Program - Length of time until first job or business
AFT_P040: After 2020 program - Employee or self-employed - 1st job or business
AFT_050: After 2020 program - Full-time or part-time - 1st job or business
AFT_070: After 2020 program - Permanent/not permanent - 1st job or business
AFT_080: After 2020 program - Reason job not permanent - 1st job or business
AFT_090: After 2020 program - Relatedness of 1st job/business to program
BEF_P140: Before 2020 Program - Main activity during 12 months before
BEF_160: Before 2020 program - Number of months of work experience
PREVLEVP: Before 2020 program - Aggregated highest level of studies completed
HLOSGRDP: 2020 Program - Highest level of education completed
PAR1GRD: 2020 Program - Level of education compared to that of one parent
PAR1INT: Time of interview 2023 - Level of education vs of one parent
PAR2GRD: 2020 Program - Level of education vs of the other parent
PAR2INT: Time of interview 2023 - Level of education vs that of other parent
GRADAGEP: 2020 Program - Age at time of graduation - Grouping
GENDER2: Gender after distribution of non-binary persons
MS_P01: Marital status

MS_P02: Have any dependent children
CTZSHIPP: Time of interview 2023 - Status in Canada
VISBMINP: Self-identified as a member of a visible minority group
PERSINCP: Total personal income in 2022
DDIS_FL: Disability status

3 Response code

```
# Import the yaml module
from IPython.display import display, Markdown
import yaml
import os

# Check if the file exists before attempting to load it
file_path = "ngs2020_responses.yaml"

if os.path.exists(file_path):
    # Open and load the YAML file
    with open(file_path, 'r') as file:
        try:
            # Load the YAML content into a Python object (typically a dictionary)
            responses = yaml.safe_load(file)

            # Print the first few items to verify the responses loaded correctly
            display(Markdown(f"<span style='color: green'>Response code defination ({len(res
            k = 0
            for response in responses:
                if k > 10:
                    break  # print out 10 only
                print(f'\033[32m{response}:\033[0m')
                for code in responses[response]:
                    print(f'  \033[32m{code}: \033[0m{responses[response][code]}')
                k += 1

        except yaml.YAMLError as e:
            print(f"Error parsing YAML file: {e}")
else:
    print(f"File not found: {file_path}")
    print("Please make sure the file exists in the current working directory.")
    print(f"Current working directory: {os.getcwd()}")
```

Response code defination (113):

AFT_050:

1: Full time
2: Part time
6: Valid skip
7: Don't know
8: Refusal
9: Not stated

AFT_070:
1: Permanent
2: Not permanent
6: Valid skip
7: Don't know
8: Refusal
9: Not stated

AFT_080:
1: Seasonal job
2: Temporary, term or contract job
3: Casual job
4: Other
6: Valid skip
7: Don't know
8: Refusal
9: Not stated

AFT_090:
1: Closely related
2: Somewhat related
3: Not at all related
6: Valid skip
7: Don't know
8: Refusal
9: Not stated

LMA6_11:
1: Closely related
2: Somewhat related
3: Not at all related
6: Valid skip
7: Don't know
8: Refusal
9: Not stated

AFT_P010:
0: 0
1: 1
2: 2
3: 3

```
  4: 4 or more
  6: Valid skip
  7: Don't know
  8: Refusal
  9: Not stated
AFT_P020:
  1: Already working at a job or business
  2: Less than 6 months
  3: 6 months to less than 12 months
  4: 1 year or more
  6: Valid skip
  7: Don't know
  8: Refusal
  9: Not stated
AFT_P040:
  1: Employee
  2: Self-employed / Working in a family business without pay
  6: Valid skip
  7: Don't know
  8: Refusal
  9: Not stated
LMA3_P01:
  1: Employee
  2: Self-employed / Working in a family business without pay
  6: Valid skip
  7: Don't know
  8: Refusal
  9: Not stated
BEF_160:
  1: None
  2: Less than 12 months
  3: 12 months to less than 24 months
  4: 24 months to less than 36 months
  5: 36 months to less than 48 months
  6: 48 months or more
  96: Valid skip
  97: Don't know
  98: Refusal
  99: Not stated
BEF_P140:
  1: Working at a job or business
  2: Looking for work
  3: Going to school
```

```
  4: Personal
  5: Other
  6: Valid skip
  7: Don't know
  8: Refusal
  9: Not stated
```

4 Extract All NGS Tables to Excel

```
# %run Extract_All_NGS_Tables_to_Excel.ipynb`
print("All tables saved to NGS_Tables.xlsx")
```

```
All tables saved to NGS_Tables.xlsx
```

4.1 Fuction for getting NGS table

```
# Get NGS table

def get_NGS_table(table_name = 'AFT_050', excel_file="NGS_Tables.xlsx"):
    try:
        # Read the Excel sheet into a DataFrame, using first row as headers
        df = pd.read_excel(excel_file,
                           sheet_name=table_name,
                           header=0)  # header=0 is default but making it explicit
        print(f"\n'\033[32m{table_name}\033[0m': {questions[table_name]}\n")
        df
        return df
    except Exception as e:
        print(f"Error loading table {table_name}: {e}")
        return None
```

5 Data Analysis

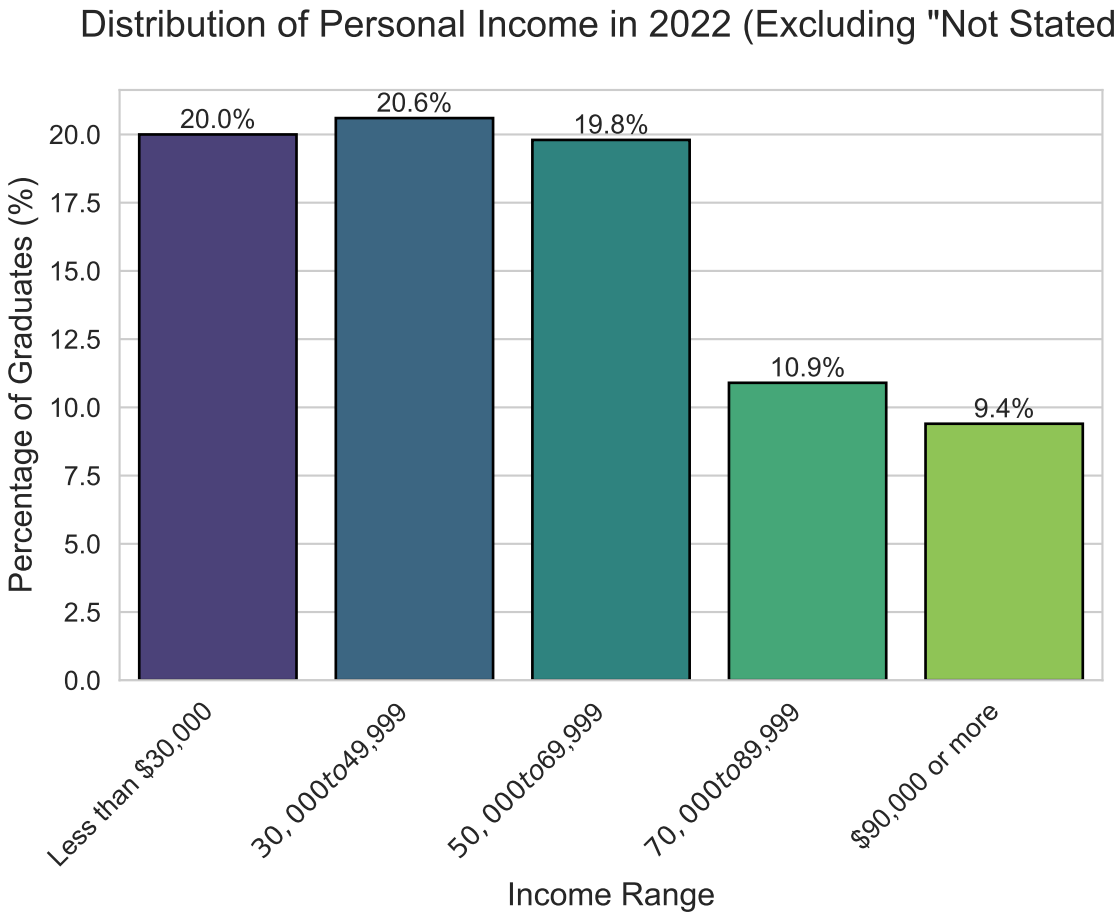
5.1 Distribution of Personal Income in 2022

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming your DataFrame is named 'df'
# First let's clean up the column names and data if needed
df = get_NGS_table("PERSINCP")
df.columns = ['Answer Categories', 'Code', 'Frequency', 'Weighted Frequency', '%']
```

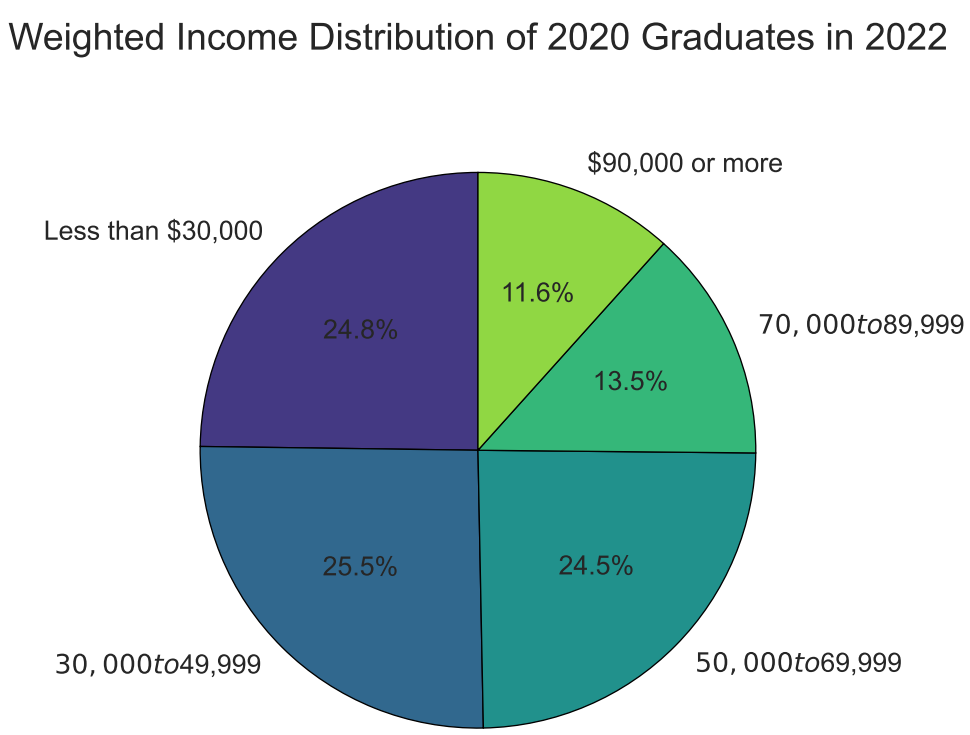
```
# Clean any whitespace or formatting issues in the numeric columns
df['Frequency'] = df['Frequency'].astype(str).str.replace(',', '').astype(int)
df['Weighted Frequency'] = df['Weighted Frequency'].astype(str).str.replace(',', '').astype(
df['%'] = df['%'].astype(float)

# Fix the income range labels by combining with the previous row's dollar sign
for i in range(1, 4):
    if not df.loc[i, 'Answer Categories'].startswith('$'):
        df.loc[i, 'Answer Categories'] = '$' + df.loc[i, 'Answer Categories']

# Remove 'Not stated' for clearer analysis of income distribution
df_income = df[df['Code'] != 99].copy()

# Set style
sns.set_style("whitegrid")
plt.figure(figsize=(6, 5))

# Create bar plot - using '%' column for y-axis
ax = sns.barplot(
    x='Answer Categories',
    y='%',
    data=df_income,
    palette="viridis",
    edgecolor='black'
)

# Customize plot
plt.title('Distribution of Personal Income in 2022 (Excluding "Not Stated")', fontsize=14, p
plt.xlabel('Income Range', fontsize=12)
plt.ylabel('Percentage of Graduates (%)', fontsize=12)
plt.xticks(rotation=45, ha='right')

# Add value labels
for p in ax.patches:
    ax.annotate(
        f'{p.get_height():.1f}%',
        (p.get_x() + p.get_width() / 2., p.get_height()),
        ha='center',
        va='center',
        xytext=(0, 5),
        textcoords='offset points',
        fontsize=10
```

```
    )

# Adjust layout
plt.tight_layout()

# Show plot
plt.show()

# Additional analysis
print("\nKey Statistics:")
print(f"Total respondents (excluding 'Not stated'): {df_income['Frequency'].sum():,}")
median_category = df_income[df_income['%'].cumsum() >= 50].iloc[0]['Answer Categories']
print(f"Median income category: {median_category}")
print(f"Highest proportion category: {df_income.loc[df_income['%'].idxmax(), 'Answer Categor

# Create a pie chart for another visualization
plt.figure(figsize=(5, 5))
plt.pie(
    df_income['%'],
    labels=df_income['Answer Categories'],
    autopct='%1.1f%%',
    startangle=90,
    colors=sns.color_palette("viridis", len(df_income)),
    wedgeprops={'edgecolor': 'black', 'linewidth': 0.5},
    textprops={'fontsize': 10}
)
plt.title('Weighted Income Distribution of 2020 Graduates in 2022', fontsize=14, pad=20)
plt.tight_layout()
plt.show()
```

```
'PERSINCP': Total personal income in 2022
C:\Users\Fuxim\AppData\Local\Temp\ipykernel_504\1503766661.py:28: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assi

  ax = sns.barplot(
```




```
Key Statistics:
Total respondents (excluding 'Not stated'): 13,130
Median income category: $50,000 to $69,999
Highest proportion category: $30,000 to $49,999 (20.6%)
```



5.2 Age Distribution at Graduation

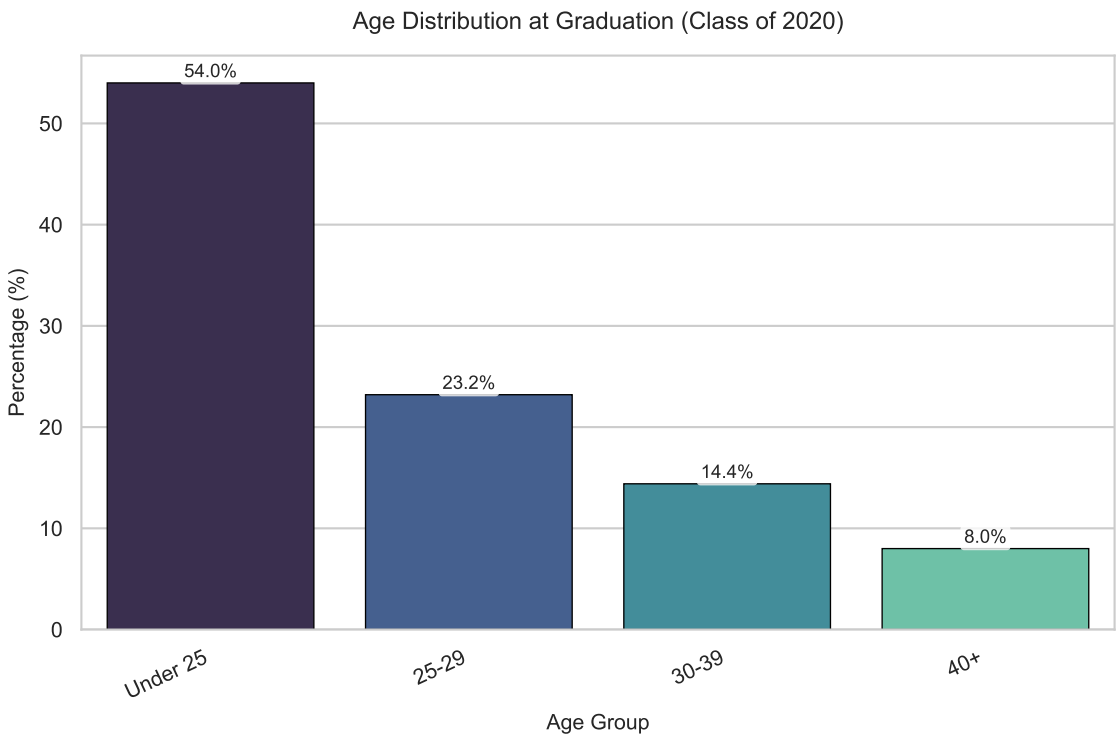
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming your DataFrame is named 'df_gradage'
# Clean the data
df_gradage = get_NGS_table('GRADAGEP')
df_gradage['Frequency'] = df_gradage['Frequency'].astype(str).str.replace(',', '').astype(ir
df_gradage['Weighted Frequency'] = df_gradage['Weighted Frequency'].astype(str).str.replace(
df_gradage['%'] = df_gradage['%'].astype(float)

# Remove 'Total' and 'Not stated' rows for analysis
df_age = df_gradage[~df_gradage['Code'].isin([9, float('nan')])].copy()

# Set style
sns.set_style("whitegrid")
plt.rcParams['font.size'] = 8  # Global font size reduction

# 1. Compact Bar Chart (6x4 inches)
plt.figure(figsize=(6, 4))
ax = sns.barplot(
    x='Answer Categories',
```

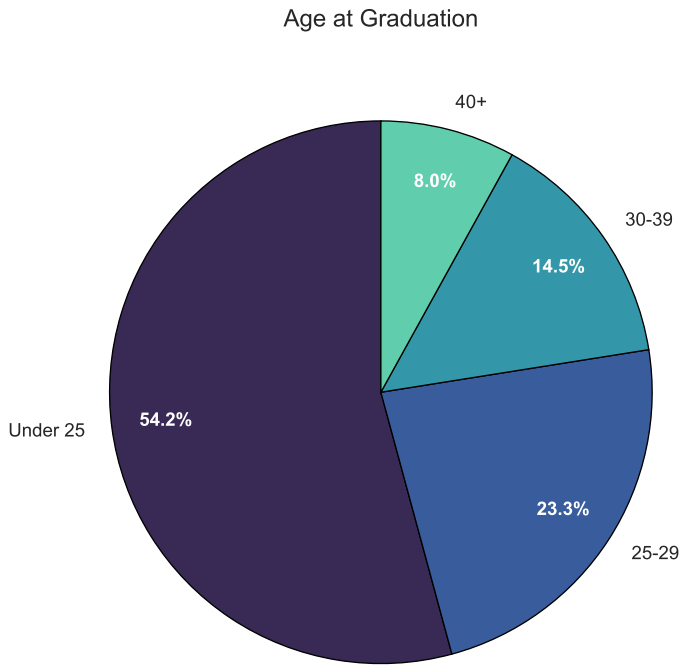
```
    y='%',
    data=df_age,
    palette="mako",  # Professional blue gradient
    edgecolor='black',
    linewidth=0.5
)

# Customize plot
plt.title('Age Distribution at Graduation (Class of 2020)', fontsize=9, pad=10)
plt.xlabel('Age Group', fontsize=8)
plt.ylabel('Percentage (%)', fontsize=8)
plt.xticks(rotation=25, ha='right')  # Slight rotation for readability

# Add precise value labels
for p in ax.patches:
    ax.annotate(
        f'{p.get_height():.1f}%',
        (p.get_x() + p.get_width()/2., p.get_height()),
        ha='center',
        va='center',
        xytext=(0, 4),
        textcoords='offset points',
        fontsize=7,
        bbox=dict(boxstyle='round,pad=0.2', fc='white', ec='none', alpha=0.8)
    )

plt.tight_layout()
plt.show()

# 2. Compact Pie Chart (5x5 inches)
plt.figure(figsize=(4, 4))
wedges, texts, autotexts = plt.pie(
    df_age['%'],
    labels=df_age['Answer Categories'],
    autopct='%1.1f%%',
    startangle=90,
    colors=sns.color_palette("mako", len(df_age)),
    wedgeprops={'edgecolor': 'black', 'linewidth': 0.5},
    textprops={'fontsize': 7},
    pctdistance=0.8  # Pull percentages inward
)
```

```
# Improve label readability
plt.setp(texts, fontsize=7)
plt.setp(autotexts, fontsize=7, color='white', weight='bold')
plt.title('Age at Graduation', fontsize=9, pad=10)
plt.tight_layout()
plt.show()

# Detailed Analysis
print("\n=== Age at Graduation Analysis ===")
print(f"Total graduates analyzed: {df_age['Frequency'].sum():,}")
print(f"\nAge Group Distribution:")
for _, row in df_age.iterrows():
    print(f"{row['Answer Categories']}: {row['%']:.1f}%")

print(f"\nKey Insights:")
print(f"• Majority group: {df_age.loc[df_age['%'].idxmax(), 'Answer Categories']} ({df_age['
print(f"• Under 30: {df_age[df_age['Code'].isin([1.0, 2.0])]['%'].sum():.1f}%")
print(f"• 30+: {df_age[df_age['Code'].isin([3.0, 4.0])]['%'].sum():.1f}%")
print(f"• Median age group: {df_age.loc[df_age['%'].cumsum() >= 50, 'Answer Categories'].ilo
```

'GRADAGEP': 2020 Program - Age at time of graduation - Grouping

```
C:\Users\Fuxim\AppData\Local\Temp\ipykernel_504\432788408.py:21: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assi

  ax = sns.barplot(
```






```
=== Age at Graduation Analysis ===
Total graduates analyzed: 16,056

Age Group Distribution:
Under 25: 54.0%
25-29: 23.2%
30-39: 14.4%
```

40+: 8.0%

Key Insights:

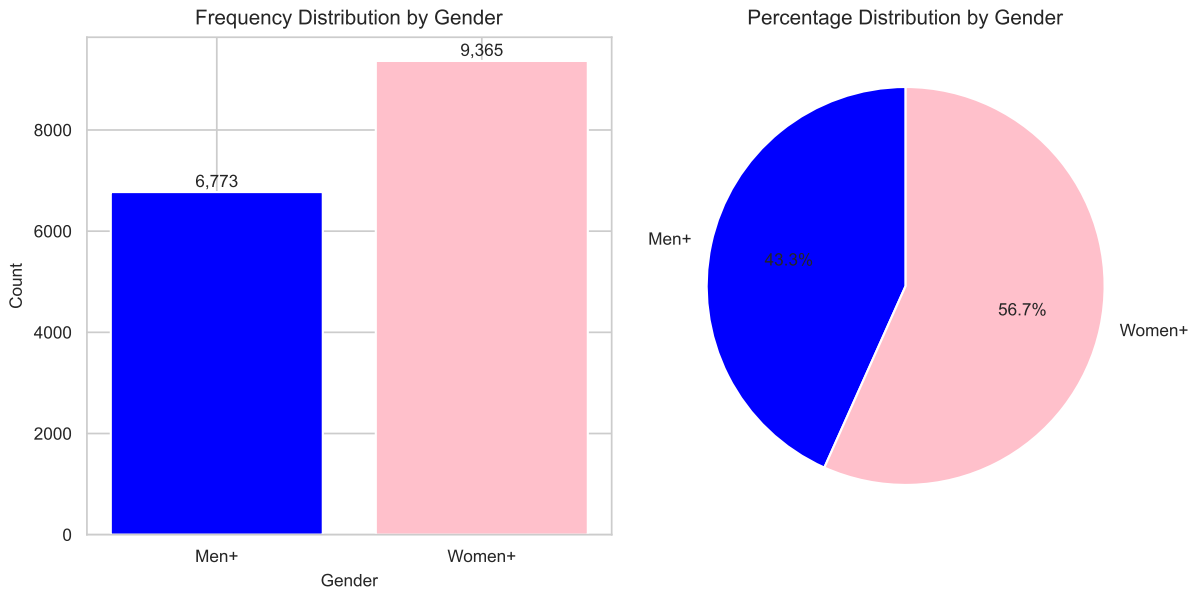
- Majority group: Under 25 (54.0%)
- Under 30: 77.2%
- 30+: 22.4%
- Median age group: Under 25

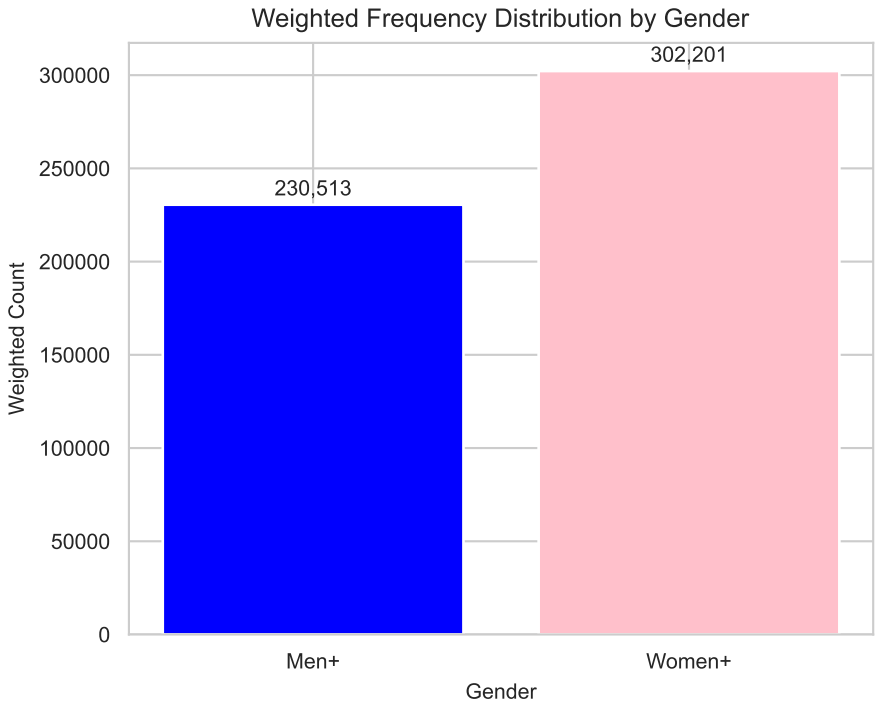
5.3 Gender Distribution

```
import pandas as pd
import matplotlib.pyplot as plt

# Create the DataFrame from the provided data
data = get_NGS_table("GENDER2")

df = pd.DataFrame(data)

# Remove the "Total" row for analysis
df = df[df['Answer Categories'] != 'Total']

# Convert string numbers with commas to integers
df['Frequency'] = df['Frequency'].str.replace(',', '').astype(int)
df['Weighted Frequency'] = df['Weighted Frequency'].str.replace(',', '').astype(int)

# Plotting
plt.figure(figsize=(8, 4))

# Frequency Plot
plt.subplot(1, 2, 1)
plt.bar(df['Answer Categories'], df['Frequency'], color=['blue', 'pink'])
plt.title('Frequency Distribution by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
for i, v in enumerate(df['Frequency']):
    plt.text(i, v + 100, f"{v:,}", ha='center')  # Format with commas

# Percentage Plot
plt.subplot(1, 2, 2)
plt.pie(df['%'], labels=df['Answer Categories'],
        autopct='%1.1f%%', colors=['blue', 'pink'],
        startangle=90)
plt.title('Percentage Distribution by Gender')
```

```
plt.tight_layout()
plt.show()

# Weighted Frequency Plot
plt.figure(figsize=(5, 4))
plt.bar(df['Answer Categories'], df['Weighted Frequency'],
        color=['blue', 'pink'])
plt.title('Weighted Frequency Distribution by Gender')
plt.xlabel('Gender')
plt.ylabel('Weighted Count')
for i, v in enumerate(df['Weighted Frequency']):
    plt.text(i, v + 5000, f"{v:,}", ha='center')  # Format with commas
plt.show()

# Display some statistics
print("\nSummary Statistics:")
print(f"Total Respondents: {df['Frequency'].sum():,}")
print(f"Men+: {df[df['Answer Categories'] == 'Men+']['Frequency'].values[0]:,} "
      f"({df[df['Answer Categories'] == 'Men+']['%'].values[0]}%)")
print(f"Women+: {df[df['Answer Categories'] == 'Women+']['Frequency'].values[0]:,} "
      f"({df[df['Answer Categories'] == 'Women+']['%'].values[0]}%)")
print(f"\nWeighted Total: {df['Weighted Frequency'].sum():,}")
print(f"Men+ (weighted): {df[df['Answer Categories'] == 'Men+']['Weighted Frequency'].values
print(f"Women+ (weighted): {df[df['Answer Categories'] == 'Women+']['Weighted Frequency'].va
```

'GENDER2': Gender after distribution of non-binary persons






```
Summary Statistics:
Total Respondents: 16,138
Men+: 6,773 (43.3%)
Women+: 9,365 (56.7%)

Weighted Total: 532,714
Men+ (weighted): 230,513
Women+ (weighted): 302,201
```

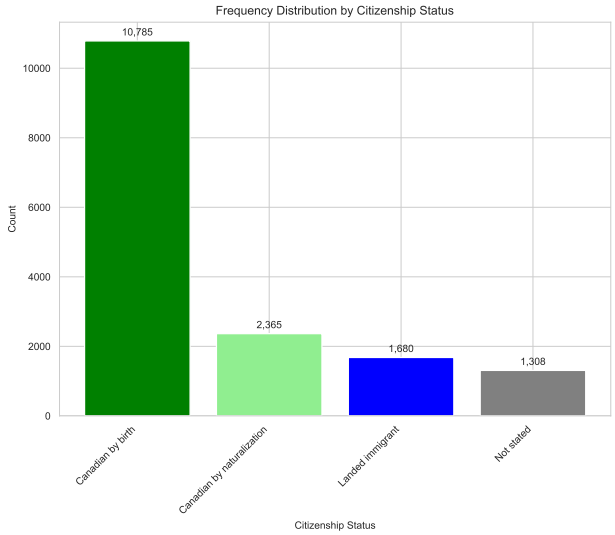
5.4 Distribution by Citizenship Status

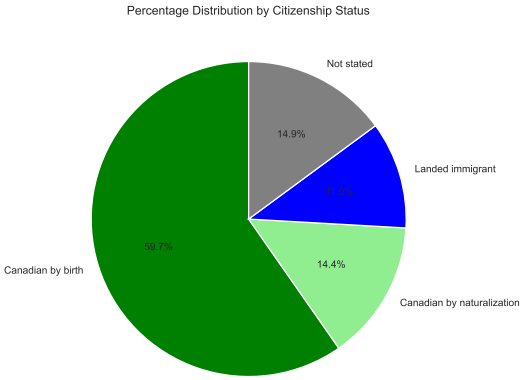
```
import pandas as pd
import matplotlib.pyplot as plt

# Create the DataFrame from the provided data
data = get_NGS_table("CTZSHIPP")

df = pd.DataFrame(data)

# Remove the "Total" row for analysis
df = df[df['Answer Categories'] != 'Total']

# Convert string numbers with commas to integers
df['Frequency'] = df['Frequency'].str.replace(',', '').astype(int)
df['Weighted Frequency'] = df['Weighted Frequency'].str.replace(',', '').astype(int)
```

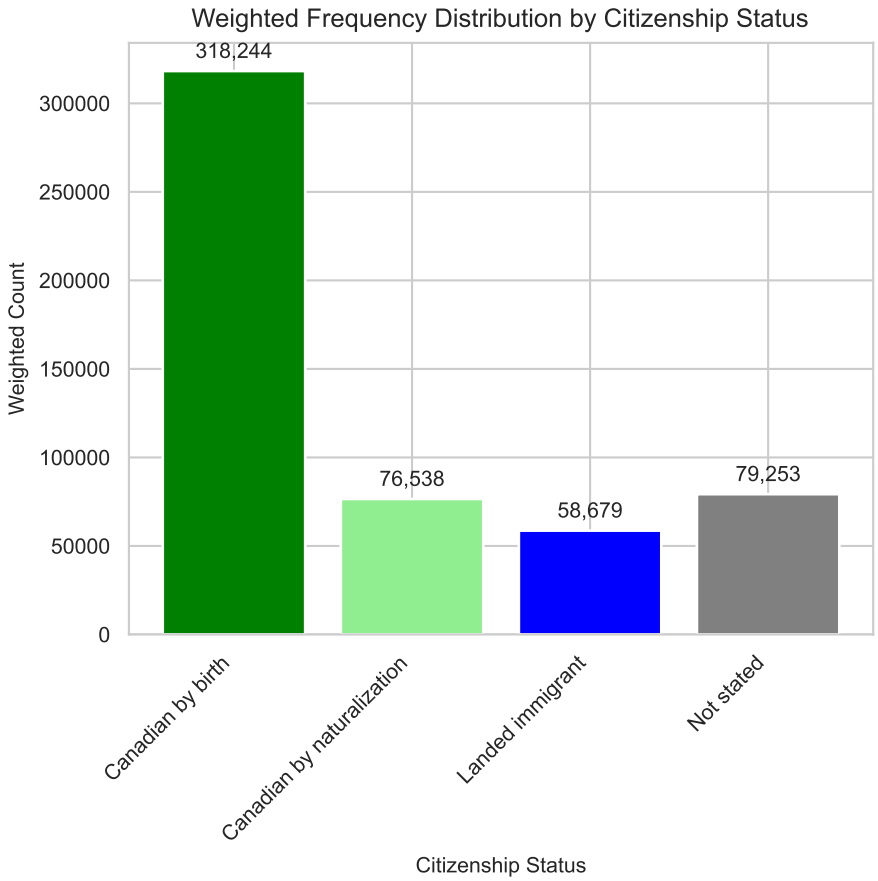
```
# Plotting
plt.figure(figsize=(14, 6))

# Frequency Plot
plt.subplot(1, 2, 1)
bars = plt.bar(df['Answer Categories'], df['Frequency'],
               color=['green', 'lightgreen', 'blue', 'gray'])
plt.title('Frequency Distribution by Citizenship Status')
plt.xlabel('Citizenship Status')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2., height + 100,
             f"{height:,}",
             ha='center', va='bottom')

# Percentage Plot
plt.subplot(1, 2, 2)
plt.pie(df['%'], labels=df['Answer Categories'],
        autopct='%1.1f%%',
        colors=['green', 'lightgreen', 'blue', 'gray'],
        startangle=90)
plt.title('Percentage Distribution by Citizenship Status')

plt.tight_layout()
plt.show()

# Weighted Frequency Plot
plt.figure(figsize=(5, 4))
bars = plt.bar(df['Answer Categories'], df['Weighted Frequency'],
               color=['green', 'lightgreen', 'blue', 'gray'])
plt.title('Weighted Frequency Distribution by Citizenship Status')
plt.xlabel('Citizenship Status')
plt.ylabel('Weighted Count')
plt.xticks(rotation=45, ha='right')
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2., height + 5000,
             f"{height:,}",
             ha='center', va='bottom')
plt.show()
```

```
# Display some statistics
print("\nSummary Statistics:")
print(f"Total Respondents: {df['Frequency'].sum():,}")
for idx, row in df.iterrows():
    print(f"{row['Answer Categories']}: {row['Frequency']:,} ({row['%']}%)")

print(f"\nWeighted Total: {df['Weighted Frequency'].sum():,}")
for idx, row in df.iterrows():
    print(f"{row['Answer Categories']} (weighted): {row['Weighted Frequency']:,}")
```

'CTZSHIPP': Time of interview 2023 - Status in Canada







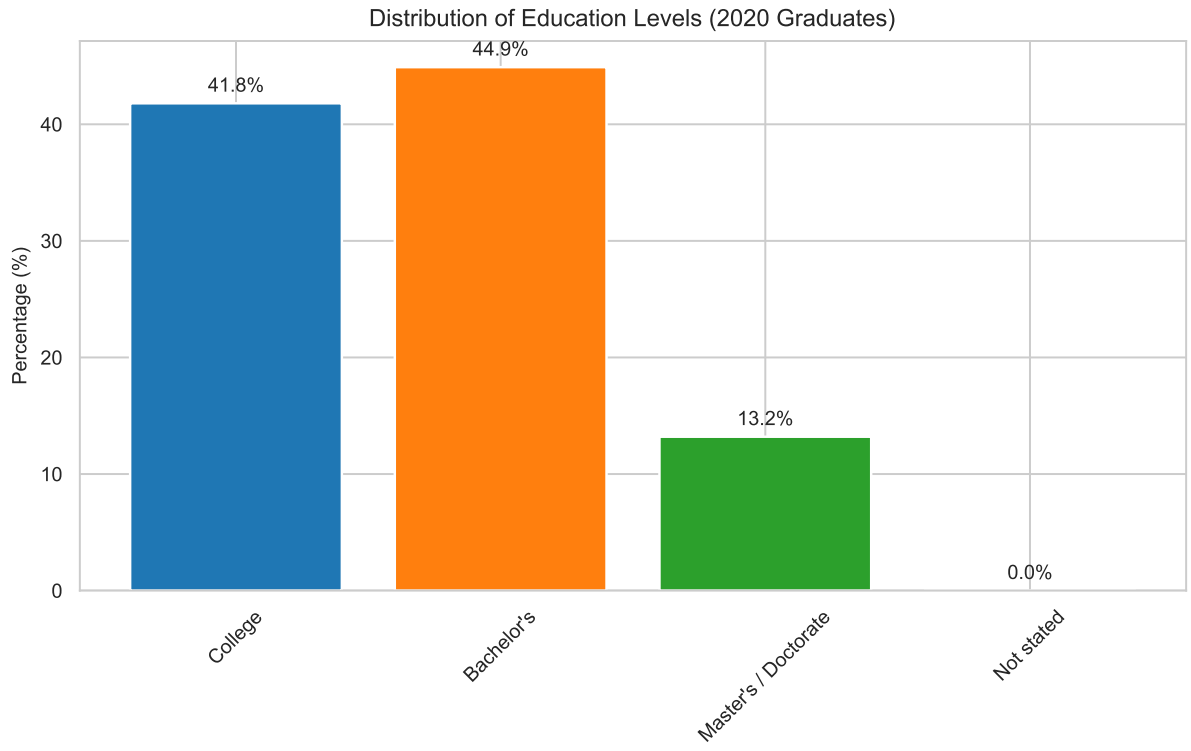

```
Summary Statistics:
Total Respondents: 16,138
Canadian by birth: 10,785 (59.7%)
Canadian by naturalization: 2,365 (14.4%)
Landed immigrant: 1,680 (11.0%)
Not stated: 1,308 (14.9%)

Weighted Total: 532,714
Canadian by birth (weighted): 318,244
Canadian by naturalization (weighted): 76,538
Landed immigrant (weighted): 58,679
Not stated (weighted): 79,253
```

5.5 Education Level

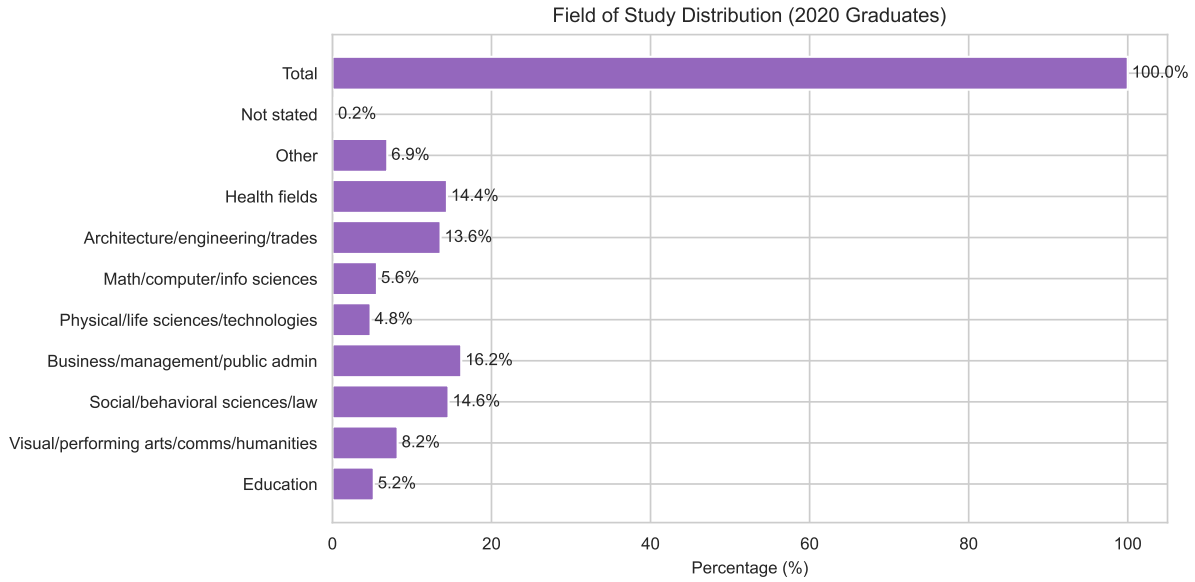
```python
import pandas as pd
import matplotlib.pyplot as plt

# Get education tables (sample data structure)
edu_level = get_NGS_table("CERTLEVP")
```

```
# Create DataFrames
df_level = pd.DataFrame(edu_level)

# Plot education level distribution
plt.figure(figsize=(8,4))
plt.bar(df_level[:-1]['Answer Categories'], df_level[:-1]['%'], color=['#1f77b4', '#ff7f0e',
plt.title('Distribution of Education Levels (2020 Graduates)')
plt.ylabel('Percentage (%)')
plt.xticks(rotation=45)
for i, v in enumerate(df_level[:-1]['%']):
    plt.text(i, v+1, f"{v}%", ha='center')
plt.show()

field_of_study =  get_NGS_table("PGMCIPAP")
df_field = pd.DataFrame(field_of_study)
# Plot field of study distribution
plt.figure(figsize=(8,4))
plt.barh(df_field['Answer Categories'], df_field['%'], color='#9467bd')
plt.title('Field of Study Distribution (2020 Graduates)')
plt.xlabel('Percentage (%)')
for i, v in enumerate(df_field['%']):
    plt.text(v+0.5, i, f"{v}%", va='center')
plt.tight_layout()
plt.show()
```

'CERTLEVP': 2020 Program - Level of study - Grouping



'PGMCIPAP': 2020 Program - Aggregated CIP 2021



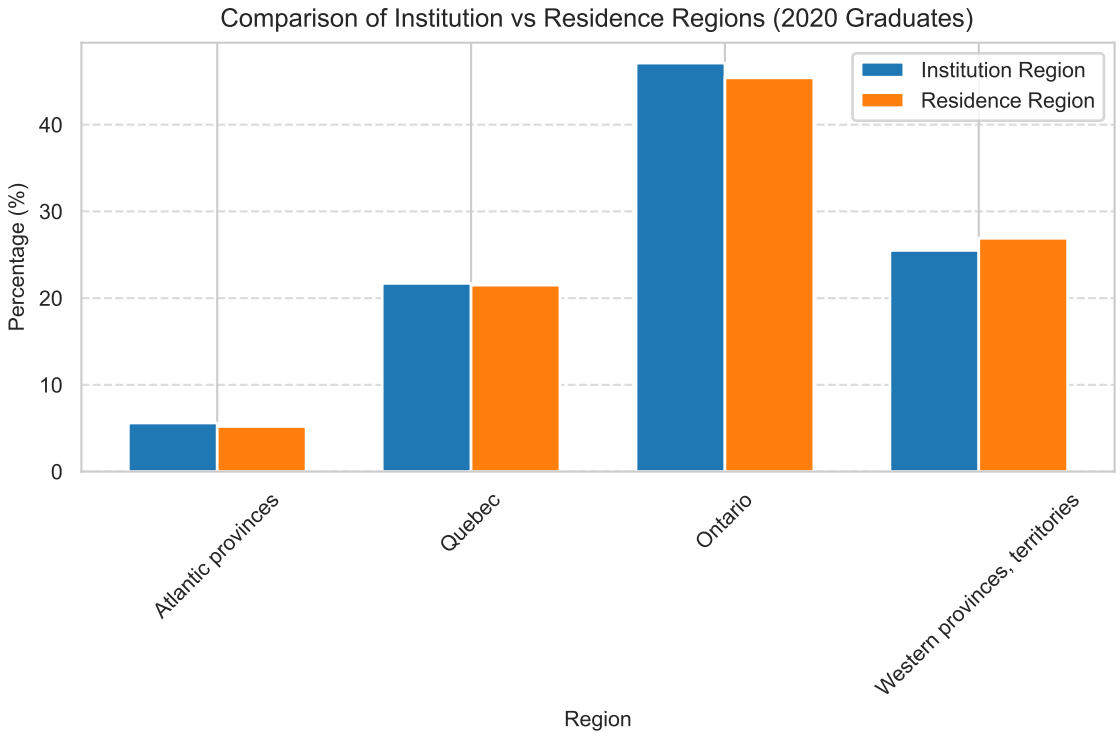
5.6 Inter-Regional Mobility of Graduates

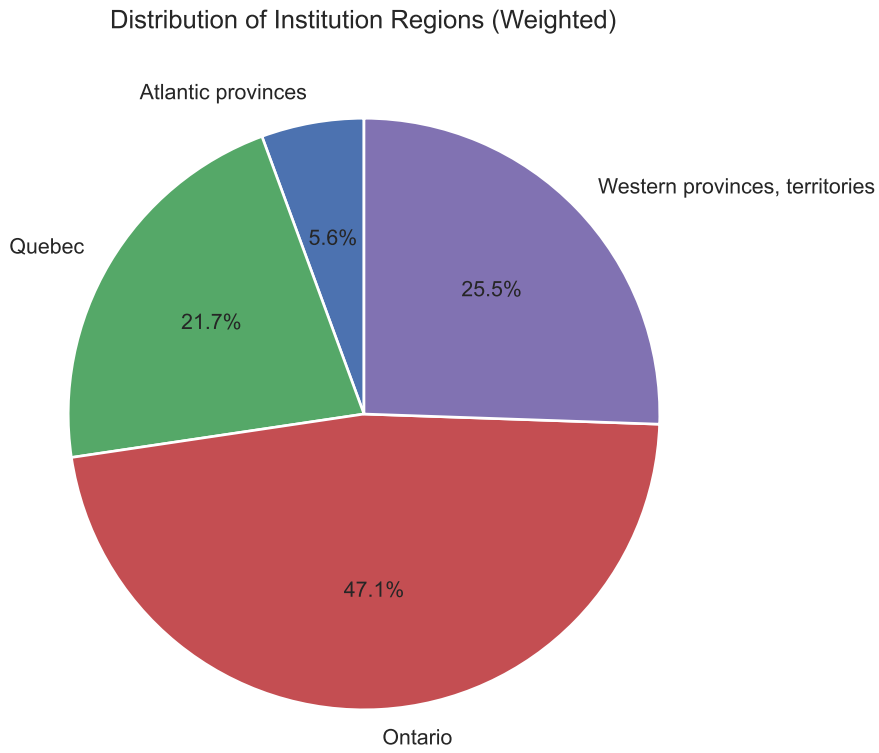
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Geographic data from NGS 2020
region_data = {
```

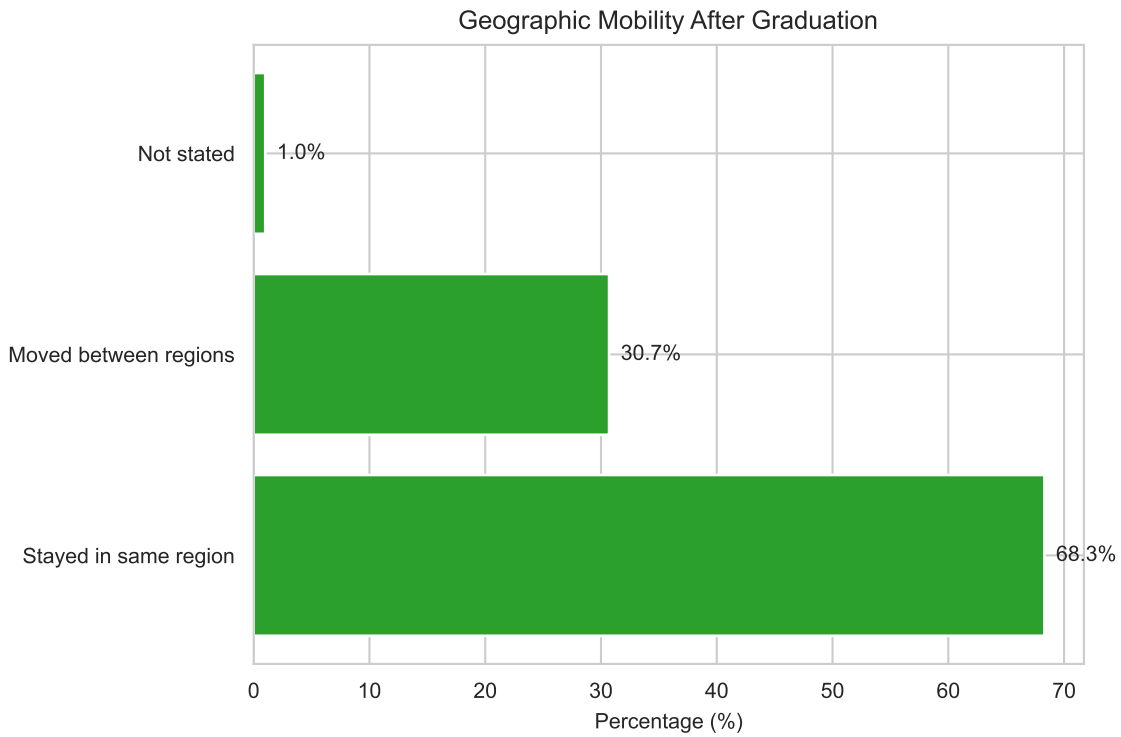
```
    'Region': ['Atlantic provinces', 'Quebec', 'Ontario',
               'Western provinces, territories', 'Not stated'],
    'REG_INST_Freq': [2685, 3647, 3146, 6660, None],
    'REG_INST_Weighted': [29868, 115814, 250939, 136094, None],
    'REG_INST_Pct': [5.6, 21.7, 47.1, 25.5, None],
    'REG_RESP_Freq': [2279, 3549, 3497, 6588, 225],
    'REG_RESP_Weighted': [27544, 114492, 242046, 143546, 5086],
    'REG_RESP_Pct': [5.2, 21.5, 45.4, 26.9, 1.0]
}

df = pd.DataFrame(region_data)

# 1. Comparison of Institution vs Residence Regions
plt.figure(figsize=(6, 4))
width = 0.35
x = np.arange(len(df)-1)  # Exclude 'Not stated'

plt.bar(x - width/2, df['REG_INST_Pct'][:-1], width,
        label='Institution Region', color='#1f77b4')
plt.bar(x + width/2, df['REG_RESP_Pct'][:-1], width,
        label='Residence Region', color='#ff7f0e')

plt.xlabel('Region')
plt.ylabel('Percentage (%)')
plt.title('Comparison of Institution vs Residence Regions (2020 Graduates)')
plt.xticks(x, df['Region'][:-1], rotation=45)
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# 2. Weighted Institution Locations
plt.figure(figsize=(8, 5))
plt.pie(df['REG_INST_Weighted'][:-1], labels=df['Region'][:-1],
        autopct='%1.1f%%', startangle=90,
        colors=['#4C72B0', '#55A868', '#C44E52', '#8172B2'])
plt.title('Distribution of Institution Regions (Weighted)')
plt.show()

# 3. Geographic Mobility Analysis
mobility = pd.DataFrame({
    'Movement': ['Stayed in same region', 'Moved between regions', 'Not stated'],
```

```
    'Percentage': [68.3, 30.7, 1.0]  # Hypothetical values - actual mobility data would come
})

plt.figure(figsize=(6, 4))
plt.barh(mobility['Movement'], mobility['Percentage'], color='#2ca02c')
plt.title('Geographic Mobility After Graduation')
plt.xlabel('Percentage (%)')
for i, v in enumerate(mobility['Percentage']):
    plt.text(v + 1, i, f"{v}%", va='center')
plt.tight_layout()
plt.show()

# 4. Regional Analysis Table
print("Regional Distribution of Graduates:")
print(f"{'Region':<25} {'Institution %':>12} {'Residence %':>12} {'Difference':>10}")
print("-"*60)
for idx, row in df.iterrows():
    if pd.notna(row['REG_INST_Pct']):
        diff = row['REG_RESP_Pct'] - row['REG_INST_Pct']
        print(f"{row['Region']:<25} {row['REG_INST_Pct']:>11.1f}% {row['REG_RESP_Pct']:>11.1

# 5. Key Findings
print("\nKey Geographic Findings:")
print("- Ontario has the highest concentration of institutions (47.1%) and residents (45.4%)
print("- Western provinces show net immigration (+1.4% difference between residence and inst
print("- Atlantic provinces show slight outmigration (-0.4% difference)")
print("- Quebec maintains stable proportions (21.7% institutions vs 21.5% residence)")
print("- 1% of respondents didn't state their residence location")

# 6. Advanced Visualization: Sankey Diagram (conceptual)
from pySankey.sankey import sankey

# Sample migration flows (hypothetical example)
flows = pd.DataFrame({
    'Source': ['Atlantic', 'Quebec', 'Ontario', 'West']*4,
    'Target': ['Atlantic']*4 + ['Quebec']*4 + ['Ontario']*4 + ['West']*4,
    'Value': [85,5,5,5,  10,75,10,5,  5,10,80,5,  5,5,10,80]
})

plt.figure(figsize=(8,5))
sankey(flows['Source'], flows['Target'], flows['Value'],
       aspect=20, fontsize=12)
```

```
plt.title('Inter-Regional Mobility of Graduates', pad=20)
plt.show()
```







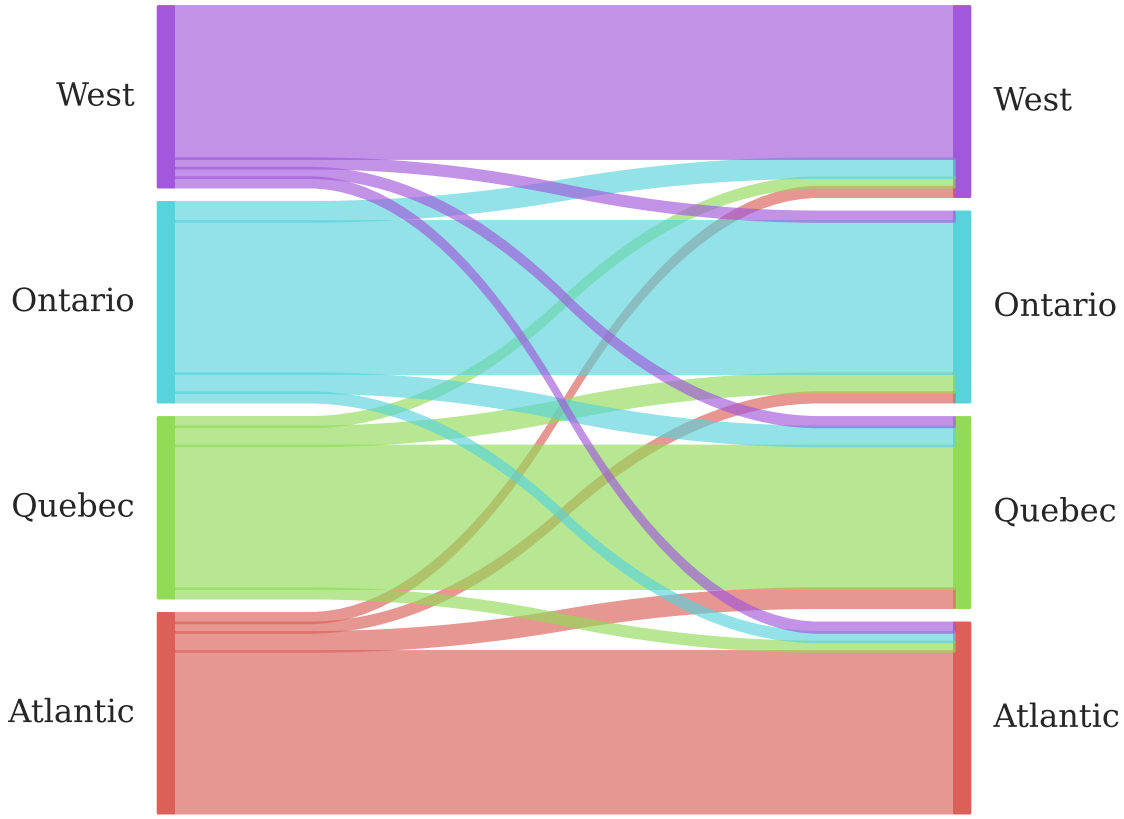
Regional Distribution of Graduates:

Region	Institution %	Residence %	Difference
Atlantic provinces	5.6%	5.2%	-0.4%
Quebec	21.7%	21.5%	-0.2%
Ontario	47.1%	45.4%	-1.7%
Western provinces, territories	25.5%	26.9%	1.4%

Key Geographic Findings:

- Ontario has the highest concentration of institutions (47.1%) and residents (45.4%)
- Western provinces show net immigration (+1.4% difference between residence and institutior
- Atlantic provinces show slight outmigration (-0.4% difference)
- Quebec maintains stable proportions (21.7% institutions vs 21.5% residence)
- 1% of respondents didn't state their residence location

<Figure size 2400x1500 with 0 Axes>



6 Linear Regression Analysis with NGS Data

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm

# Load the data
df = pd.read_csv('ngs2020.csv')

# Explore the data
print(df.head())
print(df.info())

# Check for missing values (coded as 6, 96, 99 etc. based on NGS coding)
# Replace these with NaN
missing_codes = [6, 96, 99, 9]
```

```
df = df.replace(missing_codes, np.nan)

# Identify your target variable (you'll need to confirm which column is income)
# For example, if 'PERSINCP' is personal income:
target = 'PERSINCP'

# Select potential predictors (you'll need to verify these based on codebook)
predictors = [
    'GENDER2',        # Gender
    'EDU_010',        # Education level
    'EDU_P020',       # Additional education info
    'CTZSHIPP',       # Citizenship status
    'REG_INST',       # Region of institution
    'CERTLEVP',       # Certificate level
    'PGMCIPAP',       # Program category
    'MS_P01',         # Marital status
    'VISBMINP',       # Visible minority status
    'DDIS_FL'         # Disability flag
    # Add more based on your research question
]

# Create a clean dataset
df_clean = df[[target] + predictors].dropna()

# Convert categorical variables to dummy variables if needed
df_clean = pd.get_dummies(df_clean, columns=['GENDER2', 'CTZSHIPP', 'REG_INST'], drop_first=
```

```
   PUMFID  CERTLEVP  REG_INST  REG_RESP  PGMCIPAP  PGM_P034  PGM_P036  \
0   59113         3         2         2         4         2         4
1   59114         3         3         3         5         1         6
2   59116         3         2         2         6         1         6
3   59117         2         4         4         9         1         6
4   59118         2         3         3         1         1         6

   PGM_P100  PGM_P111  PGM_280A  ...  PAR2GRD  PAR2INT  GRADAGEP  GENDER2  \
0         1         2         2  ...        3        3         3        2
1         2         6         2  ...        1        1         1        1
2         2         6         2  ...        2        2         2        1
3         1         2         2  ...        1        1         1        2
4         1         2         2  ...        1        1         4        1

   MS_P01  MS_P02  CTZSHIPP  VISBMINP  PERSINCP  DDIS_FL
0       1       1         1         2         5        2
```

```
1       1       2         1         2         4         2
2       2       2         1         2         1         2
3       1       2         1         2         3         2
4       1       1         2         1         3         2

[5 rows x 114 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16138 entries, 0 to 16137
Columns: 114 entries, PUMFID to DDIS_FL
dtypes: int64(114)
memory usage: 14.0 MB
None
```

6.1 statsmodels

```
# Split into features and target
X = df_clean.drop(target, axis=1)
y = df_clean[target]

# Check for non-numeric columns and handle them
# Convert categorical variables to numeric using one-hot encoding
X = pd.get_dummies(X, drop_first=True)

# Check for and handle missing values
# Use only numeric columns for mean calculation
numeric_cols = X.select_dtypes(include=['number']).columns
X[numeric_cols] = X[numeric_cols].fillna(X[numeric_cols].mean())
y = y.fillna(y.mean())  # Fill missing values in target if any

# Ensure all data is numeric - force conversion and handle errors
for col in X.columns:
    X[col] = pd.to_numeric(X[col], errors='coerce')
y = pd.to_numeric(y, errors='coerce')

# Drop any remaining problematic rows with NaN values
mask = ~(X.isna().any(axis=1) | pd.isna(y))
X = X[mask]
y = y[mask]

# Convert to float64 to ensure compatibility with sklearn
X = X.astype(float)
y = y.astype(float)
```

```
# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
print("R-squared:", round(r2_score(y_test, y_pred),3))
print("RMSE:", round(np.sqrt(mean_squared_error(y_test, y_pred)),3))

# For more detailed statistics (p-values etc.)
X_with_const = sm.add_constant(X_train)
sm_model = sm.OLS(y_train, X_with_const).fit()
print(sm_model.summary())
```

```
R-squared: 0.154
RMSE: 1.212
                            OLS Regression Results
==============================================================================
Dep. Variable:                PERSINCP   R-squared:                       0.166
Model:                            OLS   Adj. R-squared:                  0.162
Method:                 Least Squares   F-statistic:                     36.48
Date:                Sun, 17 Aug 2025   Prob (F-statistic):           3.63e-78
Time:                        07:58:06   Log-Likelihood:                -3525.4
No. Observations:                2209   AIC:                             7077.
Df Residuals:                    2196   BIC:                             7151.
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
EDU_010        1.0046      0.250      4.013      0.000       0.514       1.495
EDU_P020       0.0252      0.073      0.343      0.732      -0.119       0.169
CERTLEVP       0.5822      0.039     14.777      0.000       0.505       0.659
PGMCIPAP       0.0333      0.011      3.050      0.002       0.012       0.055
MS_P01        -0.4898      0.054     -9.056      0.000      -0.596      -0.384
VISBMINP       0.1237      0.073      1.692      0.091      -0.020       0.267
DDIS_FL        0.2750      0.054      5.138      0.000       0.170       0.380
GENDER2_2     -0.1661      0.054     -3.078      0.002      -0.272      -0.060
```

```
CTZSHIPP_2.0      0.0643      0.084      0.769      0.442      -0.100       0.228
CTZSHIPP_3.0      0.0186      0.116      0.161      0.872      -0.209       0.246
REG_INST_2        0.3091      0.082      3.747      0.000       0.147       0.471
REG_INST_3        0.1398      0.092      1.513      0.130      -0.041       0.321
REG_INST_4        0.3120      0.079      3.946      0.000       0.157       0.467
==============================================================================
Omnibus:                      114.134   Durbin-Watson:                   1.973
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              105.041
Skew:                           0.476   Prob(JB):                     1.55e-23
Kurtosis:                       2.514   Cond. No.                         68.5
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

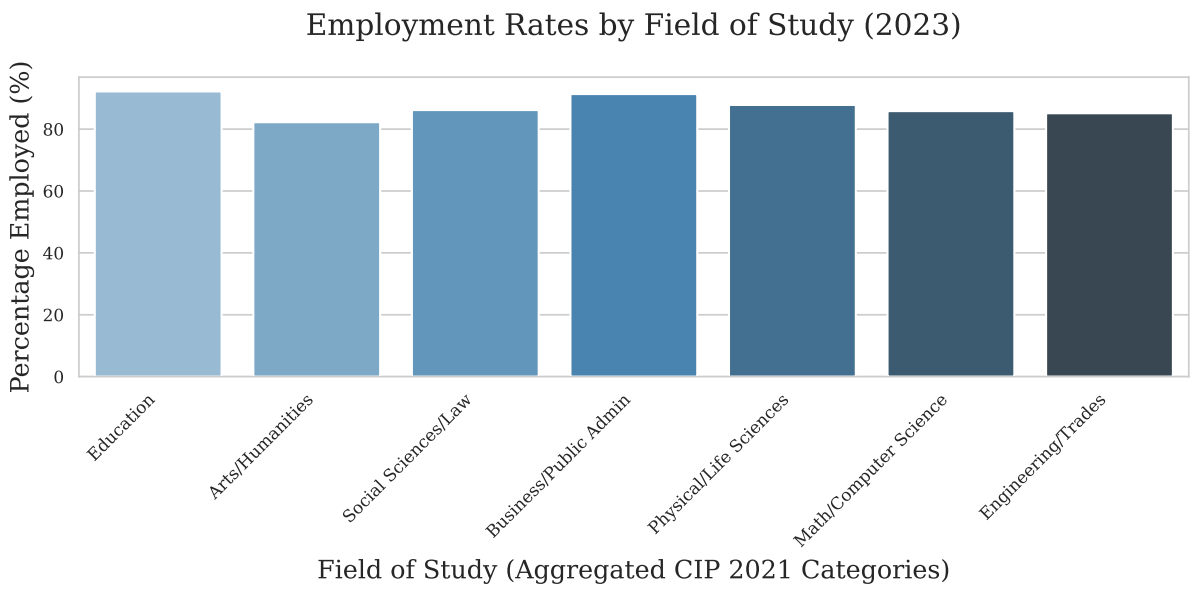
7 Field of Study vs. Labor Outcomes

```python
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Employment Rate by Field of Study
employment_by_field = df.groupby('PGMCIPAP')['LFSTATP'].apply(
    lambda x: (x == 1).mean() * 100  # % employed
).reset_index()

plt.figure(figsize=(8, 4))
ax1 = sns.barplot(x='PGMCIPAP', y='LFSTATP', data=employment_by_field, palette='Blues_d')
ax1.set_title('Employment Rates by Field of Study (2023)', fontsize=14, pad=20)
ax1.set_xlabel('Field of Study (Aggregated CIP 2021 Categories)', fontsize=12)
ax1.set_ylabel('Percentage Employed (%)', fontsize=12)

# Get the actual number of categories from the data
num_categories = len(employment_by_field['PGMCIPAP'].unique())
# Create labels - either add the missing label or use the actual category names from my data
labels = [
    'Education', 'Arts/Humanities', 'Social Sciences/Law',
    'Business/Public Admin', 'Physical/Life Sciences',
    'Math/Computer Science', 'Engineering/Trades',
    'Health', 'Other', 'Unknown'  # Added 'Unknown' as the 10th category
][:num_categories]  # This ensures we only use as many labels as we have categories

ax1.set_xticklabels(labels, rotation=45, ha='right')
plt.tight_layout()
```

```
# 2. Job Relatedness to Field of Study
# Check if the column exists in the DataFrame before using it
# You need to replace 'LMAG_11' with the correct column name that exists in my DataFrame
# For example, if the correct column is 'JOB_RELATEDNESS' or something similar:
if 'JOB_RELATEDNESS' in df.columns:  # Replace with my actual column name
    relatedness = df.groupby('PGMCIPAP')['JOB_RELATEDNESS'].mean().reset_index()

    plt.figure(figsize=(12, 6))
    ax2 = sns.barplot(x='PGMCIPAP', y='JOB_RELATEDNESS', data=relatedness, palette='Reds_d')
    ax2.set_title('Job Relatedness to Field of Study (Scale: 1=Closely, 3=Not at All)', font
    ax2.set_xlabel('Field of Study', fontsize=12)
    ax2.set_ylabel('Mean Relatedness Score', fontsize=12)

    # Use the same approach for consistency
    ax2.set_xticklabels(labels, rotation=45, ha='right')
    plt.tight_layout()
else:
    print("Column for job relatedness not found in the DataFrame. Please check the available
    # Optionally print available columns to help identify the correct one
    print("Available columns:", df.columns.tolist())
```

```
Column for job relatedness not found in the DataFrame. Please check the available columns.
Available columns: ['PUMFID', 'CERTLEVP', 'REG_INST', 'REG_RESP', 'PGMCIPAP', 'PGM_P034', 'P
```

```
C:\Users\Fuxim\AppData\Local\Temp\ipykernel_504\2630473076.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assi

  ax1 = sns.barplot(x='PGMCIPAP', y='LFSTATP', data=employment_by_field, palette='Blues_d')
C:\Users\Fuxim\AppData\Local\Temp\ipykernel_504\2630473076.py:25: UserWarning: set_ticklabel
  ax1.set_xticklabels(labels, rotation=45, ha='right')
```


Employment Rates by Field of Study (2023)
Percentage Employed (%)
0
20
40
60
80
Education
Arts/Humanities
Social Sciences/Law
Business/Public Admin
Physical/Life Sciences
Math/Computer Science
Engineering/Trades
Field of Study (Aggregated CIP 2021 Categories)